design ideas

Edited by Bill Travis and Anne Watson Swager

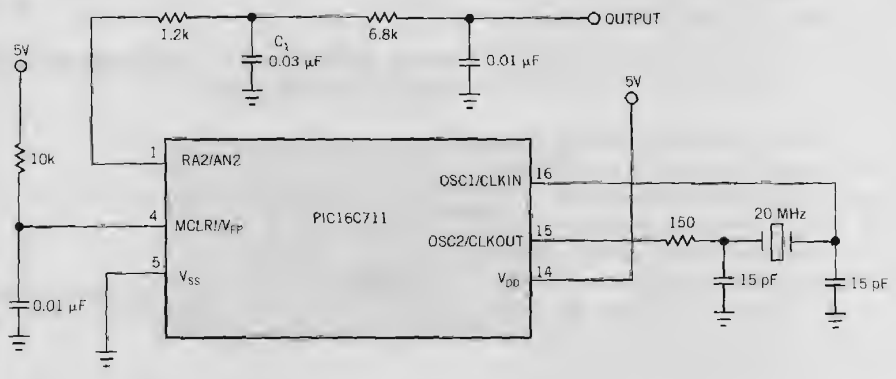
NCO technique helps μC produce clean analog signals

Steve Ploss, Veridian Corp, Wright Patterson AFB, OH

A recent Design Idea described a method for producing an analog voltage from one digital output of a μC ("Generate an analog signal with a μC," *EDN*, Oct 22, 1998, pg 108). The method involves generating a PWM output with a controlled duty cycle and filtering the switching waveform with a simple single-pole RC filter. Although this method provides an accurate dc output with 8 bits of resolution, it requires a filter with a low cutoff frequency to reduce the ripple to less than 1 LSB.

An alternative method doesn't have this problem. The circuit in **Figure 1** and the corresponding control program borrow a technique from direct digital synthesizers. The technique consists of a numerically controlled oscillator (NCO) that distributes the duty cycle as evenly as possible across the main period of the output, which is 256 clocks for both the NCO and PWM approaches. **Figure 2a** and **Figure 2b** illustrate the operation of the NCO and the PWM methods, respectively, with the duty cycle set to 10/256. **Figure 2a**'s NCO digital output has the same duty cycle as **Figure 2b**'s PWM output but distributes the duty cycle evenly across the period.

Figure 1



A numerically-controlled-oscillator (NCO) technique allows a μC and a handful of passive components to output analog signals that, after filtering, have a ripple amplitude that is constant over changes in duty cycle.

The benefit of the NCO is that the ripple amplitude after filtering is almost constant with changes in the duty cycle. In contrast, the PWM method has a ripple amplitude equal to that of the NCO approach at the lowest duty cycle, and the ripple worsens at midscale.

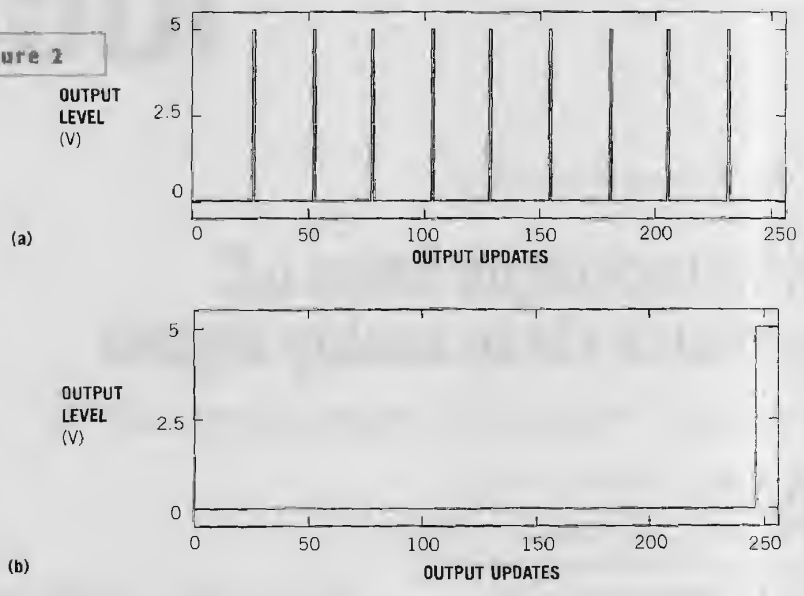
Figure 3 compares the expected output for each of the two methods using the same duty cycle as before, 10/256. A simple IIR filter that simulates a single-pole RC low-pass filter, performed the filtering. The time constant for this filter is 256 clocks. This **figure** shows that the NCO output has much lower ripple than the PWM output at this output duty cycle (**Figure 3a**). As the DAC value approaches midscale, which corresponds to a duty cycle of approximately 128/256, the PWM ripple gets progressively worse, but the NCO ripple improves by as much as a factor of 2 (**Figure 3b**).

The accompanying listing to **Figure 1** runs on the PIC16C711, but, because the routine doesn't use the ADC and the TMR0 interrupts, you can use it with other processor types. (You can download the program from *EDN*'s Web site, www.ednmag.com. At the registered-user area, go into the Software Center to download the file from DI-SIG, #2346.)

The DAC routine uses only two regis-

ters; the phase accumulator and the holding register form the DAC input value. Two additional registers provide sine-wave values to the DAC routine. On initialization, the phase accumulator sets to zero, and the DAC value sets to midscale (0x80). On each update, the value of the DAC register adds to the phase accumulator. When the phase accumulator rolls over—an event signaled by the setting of the carry bit—the circuit sets the output high for one update. The process then continues indefinitely. To generate a sine wave, the program counts the number of times it loops and every 32nd time it retrieves the next value from the lookup table. The table holds 16 values of a full cycle, so the period of the sine wave is 512 times the loop time, plus a small amount of time for the branch out of the loop. With a 20-MHz crystal, the loop time is approximately 3.5 μsec, and the sine-wave frequency is approximately 625 Hz.

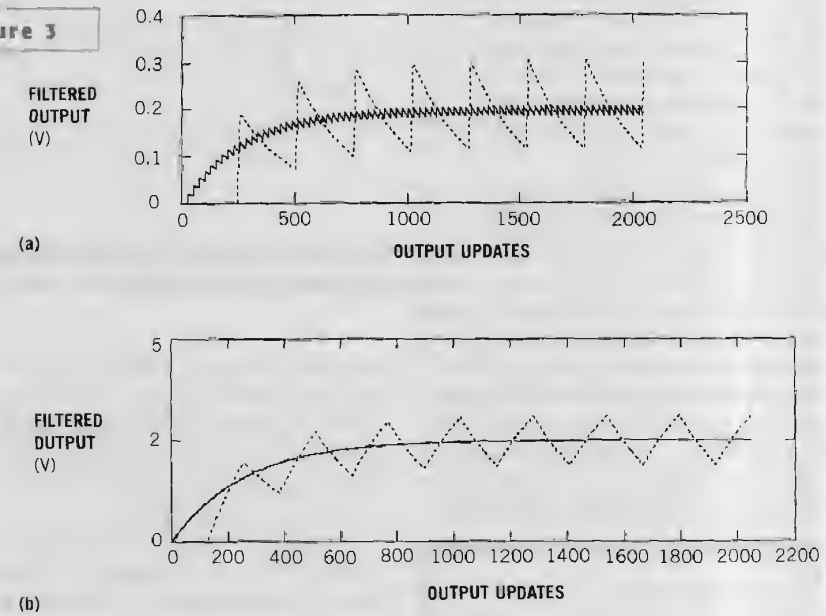
Figure 2



The NCO digital output (a) has the same duty cycle as the PWM output (b) but distributes the duty cycle evenly across the period.

To output an analog waveform, you need only to change the value of the DAC register. This change can happen at any time. The analog output almost immediately starts updating. When you use this method for your own application, remember to update the DAC output as often as possible to lower the ripple amplitude. The rate need not be precise, because you are merely setting a duty cycle—the period of which is relatively unimportant. More likely, you need to control the period of your analog waveform.

Figure 3



The output ripple that results from the numerically-controlled-oscillator (NCO) method—for example, at a duty cycle of 10/256—is much lower than that of the PWM method (a). As the DAC value approaches midscale—a duty cycle of approximately 128/256—the PWM ripple gets progressively worse, but the NCO ripple improves by as much as a factor of 2 (b).

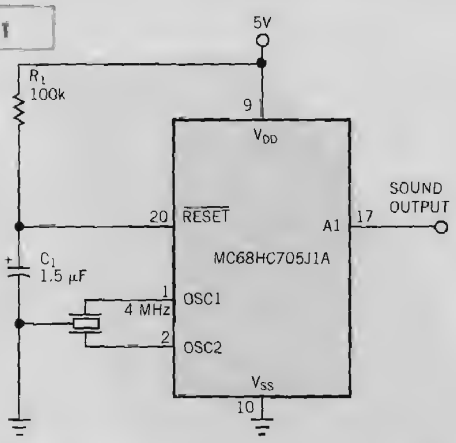
If you can afford to use a μC with a timer interrupt, it's best to use that interrupt to determine when to change to the next DAC value. The rest of the time, you can continuously update the output. If you don't want to use the timer interrupt, you can update the DAC as part of a polling loop, as shown in this example, waiting for a loop counter to reach a predetermined value before changing the DAC value. However, you may want to make sure that the branch from the polling loop always takes the same amount of time, so that the DAC-value changes occur at a constant rate. Also, if the branch is going to take a comparatively long time, consider setting the output bit to a high-impedance state for the duration of the branch. If you do not take this precaution, the output will hold the last value—whether high or low—for the duration of the branch and could produce a transient on the filter output. (DI #2346)

µC generates musical sounds

Abel Raynus, Armatron International, Melrose, MA

MANY MODERN DEVICES SEND audible signals to their operators to indicate some of the predetermined conditions or states of the system under control. To avoid annoying human sensibilities, these sounds should match the musical scale. Several chips on the market provide such sound capabilities; for example, a programmable sound generator or an ISD1016 voice messager. The circuit in **Figure 1** does not use a dedicated sound generator but rather generates sounds using software routines. The circuit uses an inexpensive MC68H705J1A µC and saves additional expense by eliminating the need for a sound chip. In **Figure 2a**, a note represents the pitch of each musical tone; **Figure 2b** shows the duration of the tone.

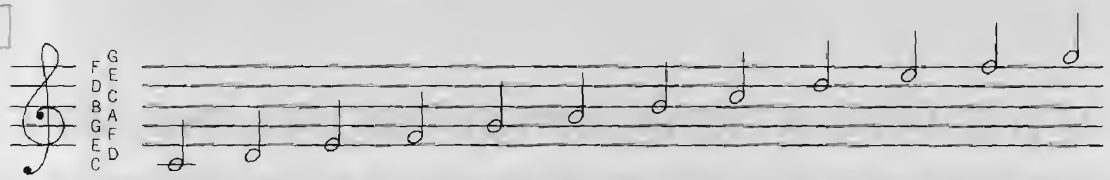
Figure 1



An inexpensive µC can provide programmable sound output without the need for expensive sound chips.

The first step in programming a sound-output system is choosing the type of sound signal: individual sounds or a section of a melody. As an example, consider a portion of the song *Jingle Bells*. Write the sequences of pitches and durations into the tables of **Figure 2** and then put them into memory in addresses aMEL and aDUR (**Listing 1**). The addresses of these tables occupy the end of the available EPROM space. The commentaries in **Listing 1** explain the structure of the subroutine MELODY. The std-jia.asm file is

Figure 2



NOTE	C	D	E	F	G	A	B	C	D	E	F	G
	DO	RE	MI	FA	SOL	LA	TI	DO1	RE1	MI1	FA1	SOL1
f, Hz	264	297	330	352	396	440	495	528	594	660	704	792
T, mSEC	3.78	3.36	3.04	2.84	2.51	2.3	2.0	1.9	1.68	1.5	1.4	1.26
1/2T, mSEC	1.89	1.68	1.52	1.42	1.26	1.14	1.01	0.94	0.84	0.76	0.71	0.63
NUMBER TO PUT INTO NR	189	168	152	142	126	114	101	94	84	76	71	63

(a)

NOTE SHAPE					
DURATION	WHOLE	HALF	QUARTER	EIGHTH	16TH
PUT INTO DR	16	8	4	2	1

(b)

A musical passage consists of pitches (a) and durations of the notes (b); you enter these values into the µC's pitch and duration registers.

the standard address list of the registers and the bytes of the μC. You can download **Listing 1** and its associated files from *EDN*'s Web site, www.ednmag.com. At the registered-user area, go into the Software Center to download the files from DI-SIG, #2336. (DI #2336).

To Vote For This Design, Circle No. 403

LISTING 1—ROUTINE FOR GENERATING MUSICAL SOUND

```
****    MUSICAL SOUND GENERATING
***********************************************
*NOLIST
$INCLUDE "std-j1a.asm"
*LIST
*I/O PORTS
snd     equ     pA1
* CONSTANTS
do      equ     189T
re      equ     168T
mi      equ     152T
fa      equ     142T
sol     equ     126T
N       equ     26T         ;number of notes in the melody
ND      equ     100T        ;number of each note repetition
aDUR    equ     ROMend-N    ;pitch duration table address
aMEL    equ     ROMend-N-N  ;note pitch table address
*VARIABLES
        ORG     RAM
NR      rmb     1           ;note pitch register
DR      rmb     1           ;pitch duration register
NDR     rmb     1           ;note duration register
memx    rmb     1
mem1    rmb     1
*INITIALIZATION
        ORG     MOR
        fcb     %00100000 ;resistor ocs
        ORG     ROM
init    rsp                 ;reset SP
        lda     #$ff
        sta     ddrA
        clr     prtA
*MAIN PROGRAM
main    jsr     dly1s       ;delay 1 sec
        jsr     melody      ;start music
        jsr     dly1s
        jsr     dly1s
        bra     main
.page
*MELODY SUBROUTINE
MELODY  clrx                ;0 -> X
m1      lda     aMEL,x      ;load the note pitch
        sta     NR          ;     into NR register
        lda     aDUR,x      ;load the pitch duration
        sta     DR          ;     into DR register
        lda     #ND         ;load the number of note
        sta     NDR         ;   repetition into NDR reg.
        stx     memx        ;save X in memory
m2      jsr     note        ;generate the note
        dec     DR
        bne     m2
        bclr    snd,prtA    ;make a pause 10ms after
        ldx     #100T       ;       each note
        jsr     DLY01x
        ldx     memx        ;reload X from memory
        cpx     #N-1        ;are there any more notes?
        blo     m3
        rts                 ;end Melody Subroutine
m3      incx                ;go to next note
        bra     m1
*****************************************************
note    bset    snd,prtA    ;start a note
        ldx     NR
        jsr     DLY01x
        bclr    snd,prtA
        ldx     NR
        jsr     DLY01x
        lda     NDR
        beq     n0
        dec     NDR
        bra     note
n0      rts                 ;end a note
.page
DLY01x  lda     #2          ;delay 0.01X ms
rep0    decA
        bne     rep0
        decx
        bne     DLY01x
        rts
*****************************************************
dly1s   lda     #10T        ;delay 1 sec
        sta     mem1
lp1     jsr     dly01s
        dec     mem1
        bne     lp1
        rts
dly01s  lda     #128T       ;delay 0.1 sec
lp2     clrx
lp3     decx
        bne     lp3
        deca
        bne     lp2
        rts
*****************************************************
*Put tables into memory
        org     aMEL
        fcb     MI,MI,MI,MI,MI,MI,MI,SOL,DO,RE,MI,
        fcb     FA,FA,FA,FA,FA,MI,MI,MI,MI,MI,RE,
        fcb     RE,MI,RE,SOL
        org     aDUR
        fcb     2,2,4,2,2,4,2,2,2,2,8,
        fcb     2,2,2,2,2,2,2,1,1,2,2,2,2,4,4
*****************************************************
        org     vectors+6
        fdb     init        ;set restart address
.end
.nolist
```

Continuity buzzer is frugal with power

Hans Krobath, EEC, Nesconset, NY

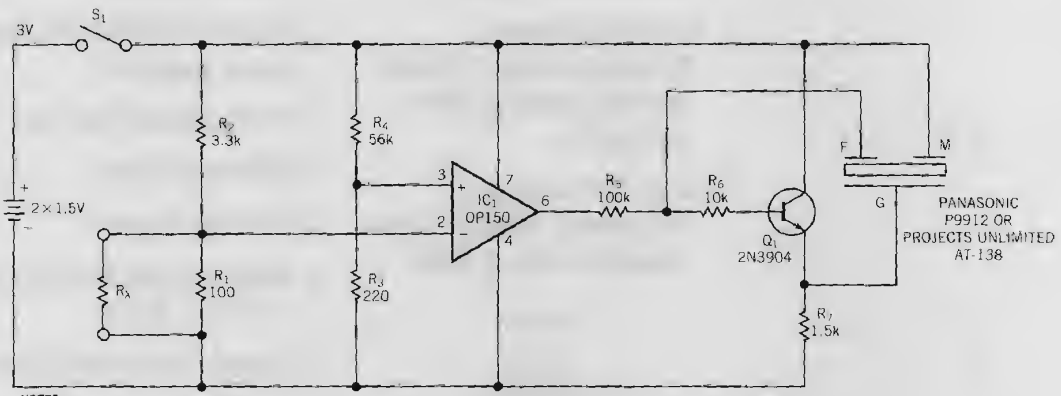
The continuity detector in **Figure 1** is based on W Dijkstra's "Fleapower circuit detects short circuits" (*EDN*, July 2, 1998, pg 122). The buzzer indicator allows you to devote full attention to making the connection without having to observe an LED. The circuit also consumes less power than Dijkstra's circuit. Power comes from two AA or AAA cells, which last for a period equal to their shelf life. Current consumption is less than 2.5 mA when the circuit de-

tects continuity and less than 1.7 mA for an open circuit. Open-circuit voltage is less than 100 mV, and short-circuit current is less than 1 mA. You can use a number of op amps for IC_1, provided that the specs indicate rail-to-rail operation with a low-voltage single supply.

The piezo oscillator driver uses only 700 μA when operating and consumes only Q_1's leakage current when it is not operating. This type of piezo transducer is a passive, resonant-feedback type, which provides high power efficiency and low-voltage operation. With R_X values greater than approximately 12Ω, the inverting input of the op amp is at a higher potential than that of the noninverting input. The resulting output is 0V plus the saturation voltage of the output stage. This output provides no bias current through R_3 and thus keeps Q_1 cut off. With R_X values lower than approximately 12Ω, the inverting input of the op amp has a potential lower than that of the noninverting input. The resulting output is 3V minus the saturation voltage of the output stage. The approximately 3V output biases Q_1 into the linear region. Q_1 and the piezo transducer, with their associated feedback, oscillate at their resonant frequency. Most transducers and the listed op amps operate with supply voltages as low as 2V. (DI #2350).

To Vote For This Design, Circle No. 404

Figure 1



This audible-signal continuity tester consumes little power and allows you to detect opens and shorts without observing an LED.

Stereo jack adds no-cost power/logic control

Gary O'Neil, IBM Microelectronics, Research Triangle Park, NC

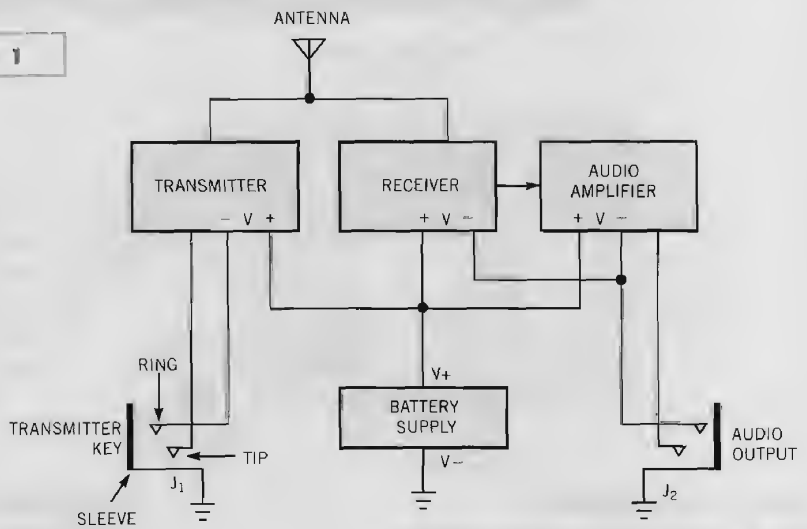
Many battery-powered devices use peripherals that require only two conductors to complete their interfaces. You can use stereo phone jacks and monaural plugs to perform power- or logic-control functions in addition to completing their required I/O connections. Monaural plugs short-circuit the ring and sleeve of stereo jacks. You can place the ring connector, normally considered a redundant ground return, into service as an spst switch. A simple wireless transceiver illustrates how you can use stereo jacks for switching with monaural plugs, stereo plugs, or both (**Figures 1** and **2**). Using the design in **Figure 1**, you can connect the battery return to the ring of stereo jacks serving one or more I/O devices. The circuit in **Figure 2** connects the returns of individual circuits to the ring.

With this scheme, power control becomes automatic with the plugging and unplugging of devices using monaural plugs, by virtue of the ring-to-sleeve short circuit. Three examples illustrate the principles of operation. In the first example, the transciever comprises a transmitter, a receiver that contains an audio amplifier, a battery supply for power, and two stereo phone jacks (J_1 and J_2) for transmitter modulation and audio output, respectively (**Figure 1**). J_1's tip con-

nection carries the modulation input to the transmitter. This input could be a keying line for CW (continuous modulation, for Morse code); a digital modulation interface; or an analog input, such as in a wireless microphone. The output of the audio amplifier (demodulated digital data, control tones, voice, music, or other) connects to the tip of J_2.

With no peripheral attached, the negative side of the battery floats, with no return to ground. Therefore, the transceiver receives no power. The negative side of the battery connects to the rings of all jacks you want to use for power-on/off control. Inserting a telegraph key, digital-modulation device, or microphone into J_1 via a monaural plug connects the negative side of the battery to ground via the ring-to-sleeve short circuit, and the transceiver turns on. Plugging an earphone or speaker into J_2 via a monaural plug completes an alternate, or redundant, ground return for the battery. Thus, inserting a plug into either jack completes the power path to all the transceiver circuits.

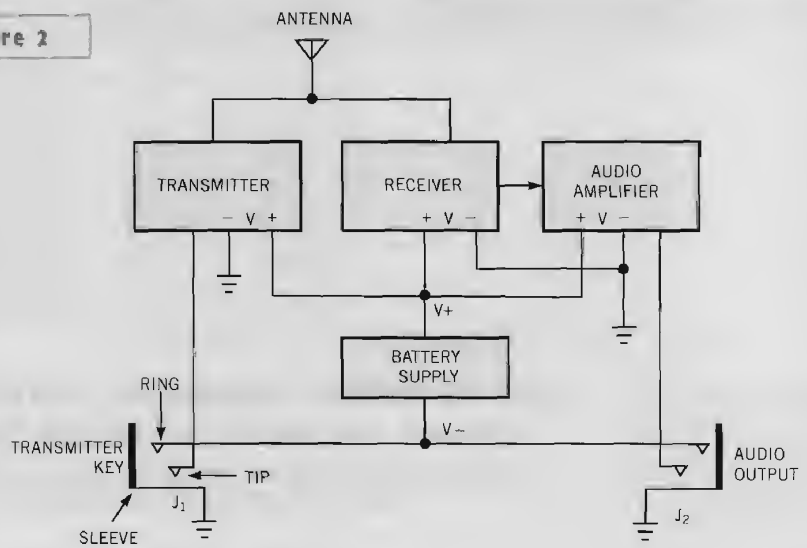
Figure 1



Inserting any monaural plug completes the battery's ground-return path, thus enabling all circuits in the transceiver.

In another example, you can selectively isolate and enable circuits using stereo jacks to maximize power conservation. When you plug a peripheral into J_1, the circuit in **Figure 2** enables only the transmitter circuit. When you plug a peripheral into J_2, the circuit enables only the receiver and audio-amplifier circuits. The return paths of the individual circuit functions float and connect to the ring of one or more stereo jacks associated with a peripheral that requires those circuit functions to operate. The battery permanently connects to the same ground reference as the sleeve of all jacks. You need both the receiver and audio amplifier to operate during the receive function; they share a common return path with the ring of J_2. You enable these circuits by inserting an appropriate peripheral (speaker or headphone) into J_2 via a monaural plug.

Figure 2



The absence of a peripheral plug breaks the ground return for either the transmitter or the receiver/audio-amplifier portion of the transceiver.

In the final example, you can further conserve power in specialized cases featuring remote on/off control, such as for transmitter on/off keying (**Figure 2**). You use insert a stereo plug into J_1 to remotely locate the ring/sleeve connection. The ring/sleeve path to the negative side of the battery is incomplete until a peripheral switch or a key closes. You can define a ring/sleeve, tip/sleeve, or ring/tip/sleeve short circuit and use it to complete the desired electrical connections. In this example, the circuit consumes power only when you enable the transmitter; thus, you have maximum control of battery resources. (DI #2355).

To Vote For This Design,
Circle No. 405

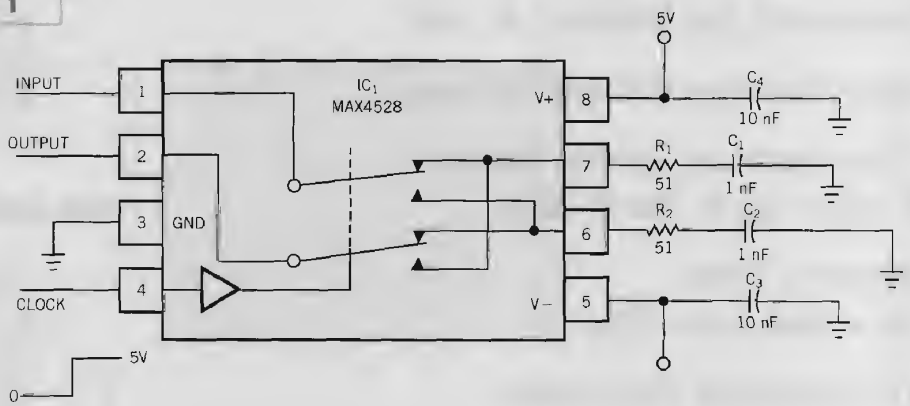
S/H circuit minimizes aperture

John Guy, Maxim Integrated Products, Sunnyvale, CA

Conventional sample-and-hold (S/H) circuits use one hold capacitor that charges during the track phase and disconnects during the hold phase. The voltage that the capacitor holds usually drives an A/D converter that operates synchronously with the S/H control signal. This approach can sometimes place excessive demands on the S/H circuit's bandwidth and settling capabilities. You can improve performance by using two hold capacitors to implement continuous sampling (**Figure 1**). One capacitor or the other is always sampling the input signal, and the output is always the held value. A phase-reversal switch (IC_1) interconnects the input, output, and hold capacitors.

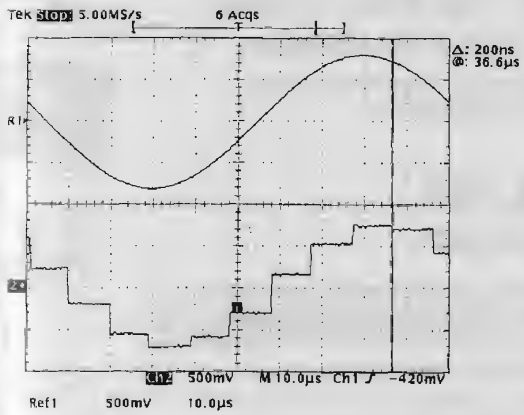
When the control signal (Clock) is low, the input connects to the C_2 hold capacitor via R_2. C_2 and the on-resistance of the switch form a 160-nsec time constant, providing ample time to charge the capacitor. When Clock goes high, C_2 connects to the output via the lower switch, and C_1 connects to the input. Many factors affect the performance of this circuit. Droop rate on the hold voltage, for example, is a strong function of the output-load impedance. If the A/D-converter load is excessive, you should add a buffer amplifier at the output of the S/H circuit. The hold capacitors should be low-dielectric-absorption types, and the phase-reversal switch should specify low charge injection and make-before-break timing. **Figure 2** (input and output at 100k samples/sec) and **Figure 3** (detail at 400k samples/sec) show good performance with the values shown. The high-speed transitions exhibit little overshoot or ringing. (DI #2354).

Figure 1



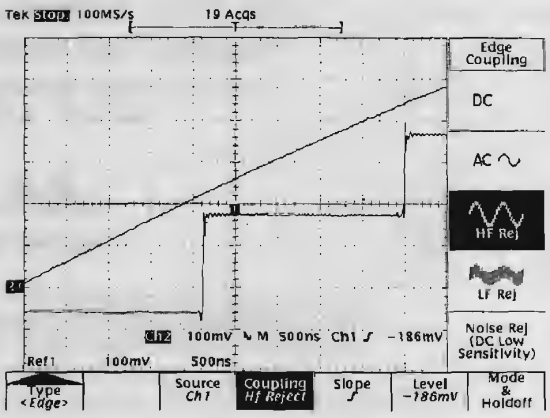
A simple phase-reversal switch uses two hold capacitors to implement a continuous-sampling sample/hold circuit.

Figure 2



At a 100k-sample/sec sampling rate, the circuit in Figure 1 provides excellent hold accuracy.

Figure 3



The S/H circuit in Figure 1 exhibits low ringing and overshoot characteristics when sampling at 400k samples/sec.